TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Vizualizace technologického procesu kontrolovaného řídicím systémem

Visualization of technologic process controlled by controll system

Bakalářská práce

Autor: **Petr Gašparík**
Vedoucí práce: Ing. Jiří Bažant
Konzultant: Ing. Jaroslav Vlach

V Liberci 18. 5. 2007

Zde vložte originál zadání

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Zde bych rád poděkoval svému vedoucímu práce, Ing. Jiřímu Bažantovi za podněty a připomínky k vypracování této bakalářské práce, dále bych rád poděkoval pánům Josefovi Havlíčkovi a Ing. Jaroslavu Vlachovi z podniku Preciosa a.s., za rady při realizaci praktické části práce.

Anotace

Námětem této bakalářské práce byla potřeba ověřit použitelnost OPC serveru pro komunikaci s řídicím systémem pro oddělení řídicích systémů podniku Preciosa a.s.

K naprogramování testovací aplikace, zobrazující komunikovaná data z OPC serveru bylo využito vývojového prostředí LabVIEW 7.1. Popsány jsou základní využité funkce a pojmy. Jsou zde vyloženy základní pojmy technologie OLE, jejíž nadstavbou je standard OPC, sloužící ke komunikaci s řídicími systémy a zařízeními průmyslové automatizace. Pro utvoření základní představy o standardu OPC a jeho použité specifikaci je zhruba popsán princip této technologie. Dále je popsán použitý OPC server, jeho vlastnosti a jeho ovládání.

V další kapitole je vyložena tvorba testovací aplikace. K navázání komunikace mezi aplikací a serverem byli využity nástroje z knihovny funkcí DataSocket a vlastní vytvořené funkce pracující na jejich principu.

V hrubých rysech je popsáno rozhraní RS232 pro komunikaci řídicího systému a počítače. Součástí kapitoly je i popis driveru, se kterým má být porovnán OPC server.

Poslední kapitola se zabývá právě rozdíly, výhodami a nevýhodami komunikace s řídicím systémem pomocí samotného driveru v kontrastu s řešení využívajícím OPC server.

Abstract

The Theme of this bachelor´s work was requirement of controll system department of Preciosa a.s. company, to verify usabiliti of OPC server in comunication with controll system.

For displaying comunicated data was developed test application in developing environment LabVIEW 7.1. There is discribed used fundamental functions and notions. Fundamental notions of OLE technology are discibed too. Upgrade of OLE is OPC, used for cominication with controll systems and industry automotion devices.

To made a fundamental idea of OPC standard and it´s specifications is described principle of it. Along is described used OPC server, his properties and controllship.

IN next chapter is explained development of test application. To link up developed application and server were used functions of DataSocket library and own developed functions using principle of DataSocket. There was described comunicating interface RS232 in common impression. In part of this chapter is described driver, which would be compared with OPC server.

In last chapter are discussed advantages and disadvantages of both comunication solutions

Obsah

Úvod..... 9
1 LabVIEW..... 10
1.1 Historie..... 10
1.2 Struktura LabVIEW..... 11
1.2.1 Front Panel (FP)..... 11
1.2.2 Block Diagram (BD)..... 11
1.2.3 Datové typy jazyka G..... 12
1.2.4 Proměnné a konstanty v prostředí LabVIEW..... 13
1.2.5 Řídící struktury..... 13
1.2.4 Vizualizace..... 14
1.3 Prostředky pro komunikaci..... 14
1.3.1DataSocket..... 15
1.3.2 ActiveX..... 16
2 OPC..... 18
2.1 Princip OPC..... 18
2.1.1 OPC Specifikace..... 19
2.1.2 OPC Server..... 20
2.1.3 OPC klient..... 20
2.2 Použitý OPC server..... 21
2.2.1 Instalace Serveru..... 21
2.2.2 KEPServerEx..... 22
3 Tvorba aplikace..... 25
3.1 Komunikace..... 25
3.1.1 DataSocket VytvorRef..... 25
3.1.2 DataSocket Pripoj..... 25
3.1.3 DataSocket ObnovData..... 26
3.1.4 DataSocket Odpoj..... 26

3.1.5 DataSocket UvolniRef..... 26
3.2 Převod dat..... 26
3.2.1 DataSocket CtiVariant..... 27
3.2.2 DataSocket OPCCas..... 27
3.2.3 Buffer..... 28
3.3 Hlavní program..... 28
4 Rozhraní průmyslové komunikace..... 31
4.1 Rozhraní RS232..... 31
4.1.1 Fyzické parametry..... 31
4.1.2 Přenosové parametry..... 31
4.1.3 Komunikace se zařízením..... 32
4.2 Driver RS232..... 32
4.2.1 Vlastnosti driveru..... 33
4.2.2 Princip driveru..... 33
4.2.3 Doporučení..... 33
5 Porovnání OPC serveru a driveru..... 35
5.1Přenosové vlastnosti..... 35
5.1.1 Rychlost přenosu..... 35
5.1.2 Velikost adresovatelného prostoru..... 36
5.2 Ostatní vlastnosti.....37
5.2.1 Přehlednost..... 37
5.2.1 Srozumitelnost..... 37
5.2.3 Použitelnost..... 37
Závěr..... 39
Seznam použitých zkratek..... 40
Použitá literatura a informační zdroje..... 41

Úvod

Hlavním cílem práce bylo zjistit, zda je použití OPC serveru vhodné pro vizualizaci dat z řídicího systému. Pokud ano, tak najít takový, který by byl použitelný pro zadaný řídicí systém. Dále zjistit výhody či případné nevýhody tohoto řešení v porovnání s driverem řídicího systému a tyto následně vyhodnotit.

V první části práce se zevrubně seznámíme s vývojovým prostředím LabVIEW, s jeho historií, strukturou a prostředky pro vizualizaci. Větší pozornost bude věnována komunikačním prostředkům.

Na to zvolna navazuje ve druhé části výklad principů a standardů OPC serveru. Popsán je použitý OPC server a nezbytnosti pro jeho bezproblémový provoz.

Následuje ověření využitelnosti OPC serveru k vizualizaci dat. K tomu má sloužit vytvoření aplikace v prostředí LabVIEW.

Čtvrtá část práce se zabývá okrajově rozhraními průmyslové komunikace, hlavně těmi, které podporuje použitý řídicí systém.

V poslední části je srovnání OPC serveru a driveru, který k datům řídicího systému přistupuje přímo.

Závěrem jsou posouzeny jednotlivé výhody a nedostatky obou řešení, jejich přínos a možnosti využití.

1 LabVIEW

LabView (Laboratory Virtual Instruments Engineering Workbench) neboli volně přeloženo do češtiny „laboratorní pracoviště virtuálních nástrojů“.

Vývojové prostředí LabVIEW, dále jen LV, je produktem americké firmy National Instruments (NI)[1], která je průkopníkem a největším výrobcem v oblasti virtuální instrumentace, technické disciplíny, která v nyní zažívá veliký rozkvět snad ve všech odvětvích průmyslu, od chemického přes automobilový až po letecký.

Hlavní myšlenkou virtuální instrumentace, je nahradit prostorově a finančně náročný hardware sofwarovým řešením, které je samozřejmě snadno a rychle re-konfigurovatelné, což je u realizace skutečnými nástroji, potažmo součástkami, často velice nákladné nebo přímo nemožné.

1.1 Historie

V roce 2006 oslavilo LV 20 let od svého uvedení na trh, současně s tím byla vydána nová verze toho programu s označením 8.20. Vývoji tohoto programovacího prostředí předcházelo v roce 1983 vyrobení desky GPIB, první z produkce NI. Jeden ze tří zakladatelů této úspěšné firmy, Jeff Kodosky, ve spolupráci s Texaskou univerzitou, započal vývoj grafického rozhraní, které by bylo nenáročné na ovládaní, jelikož v té době profese inženýra nevyžadovala znalost pokročilého programování. Vycházelo se z předpokladu, že technik, který je schopen zapsat své poznatky a požadavky do blokového schématu, by měl být schopen ovládat i toto vývojové prostředí.

Konečným produktem bylo vývojové prostředí pro Apple Macintosh, které místo klasického textového programování umožňovalo tvořit programy graficky, více intuitivně a samozřejmě i rychleji. Program obsahoval a obsahuje různé funkce, reprezentované ikonkami, které lze vzájemně spojovat virtuálními vodiči a výsledky pak vykreslovat např. do grafů. Rok po uvedení LV, byla uvedena verze pro prostředí DOS, pod názvem Lab Windows.

1.2 Struktura LabVIEW

Programovací jazyk, užívaný v LV se nazývá "G". Je to grafický programovací jazyk, ve kterém se prakticky nepíše, ale kreslí. Ze stejné rodiny programovacích jazyků je např. Simulink, který je součástí prostředí MATLAB od firmy Math Works.

Program napsaný v LV je nazýván Virtual Instrument (dále VI), neboli virtuální přístroj. Takový VI lze samozřejmě použít i jako podprogram, tedy subVI, v jiném virtuální přístoji.

1.2.1 Front Panel (FP)

Tvorba VI se skládá ze dvou částí. První je vytvoření tzv. front panelu (viz Obr. 1-1), což je vlastní uživatelské rozhraním umožňujícím čtení dat, která jsme například naměřili.

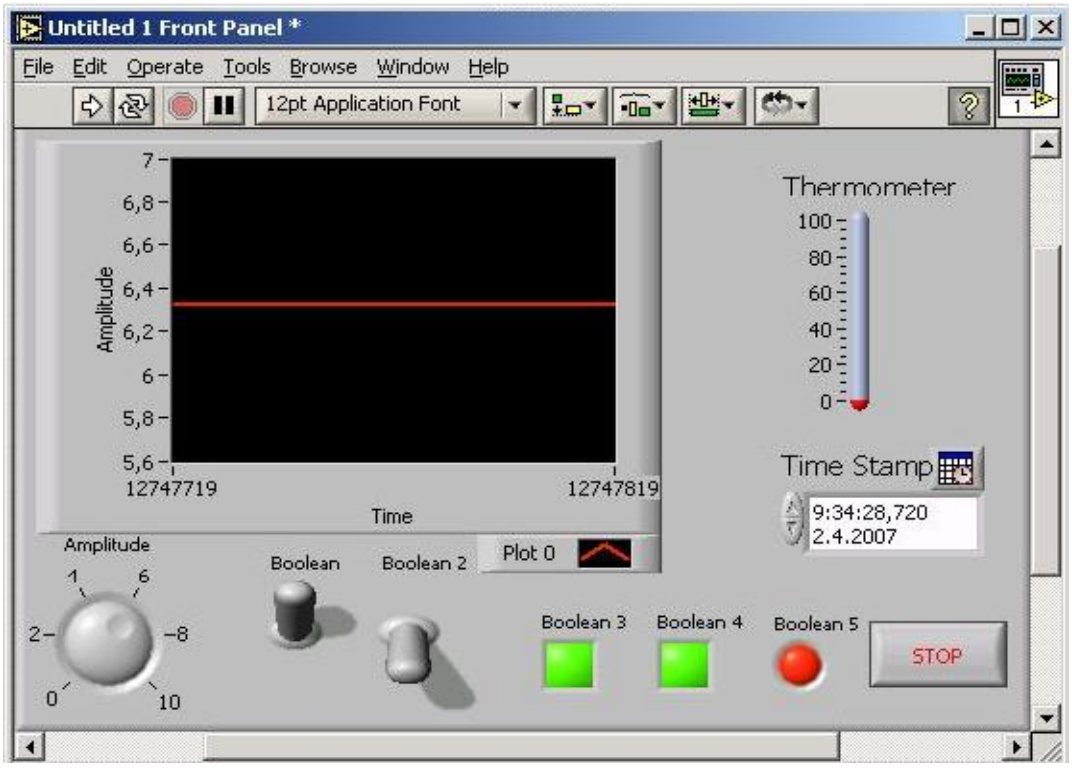
Ta lze reprezentovat různými způsoby v závislosti na formátu dat. Tuto funkci zajišťují **indicators**. Samozřejmě je také možný zápis dat a tím ovládaní daného VI, k čemuž slouží **controls**.

Všechny ovládací a indikační prvky lze jednoduše umisťovat na FP přetažením libovolného prvku z paletového menu, které se aktivuje pravým kliknutím myši. Po vytvoření uživatelského rozhraní přistoupíme k tvorbě druhé části VI.

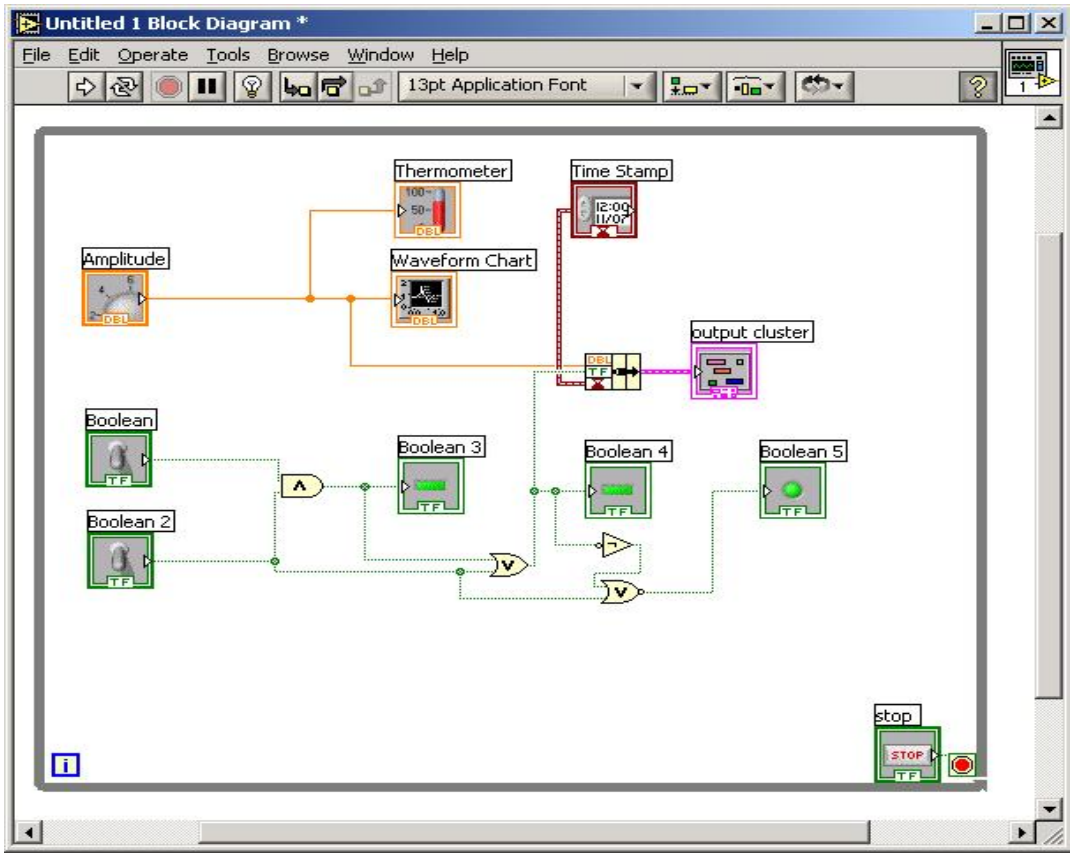
1.2.2 Block Diagram (BD)

Druhou částí VI je blokový diagram (viz Obr. 1-2), který obsahuje všechny funkce, proměnné, konstanty, cykly a subVI´s potřebné k běhu programu. Stejným způsobem jako do FP můžeme umisťovat do BD funkční bloky tzv. terminály a uzly. Potom je třeba pospojovat jednotlivé terminály tak, aby zamýšlené funkce správně fungovali.

Ovládání programu je možné jak z FP, tak z BD. Spustit program můžeme buď jednorázově (**run**) nebo v cyklu (**run continuously**). Standardně lze běh vykonávaného programu zastavit (**stop**) nebo ho přerušit (**pause)** a v BD je také možnost program krokovat. Ovládání těchto a dalších akcí se nachází na hlavní (horní) liště okna.


Untitled 1 Front Panel *
File Edit Operate Tools Browse Window Help
12pt Application Font
Amplitude
Time
12747719
12747819
Thermometer
Time Stamp
9:34:28,720
2.4.2007
Amplitude
Boolean
Boolean 2
Plot 0
Boolean 3
Boolean 4
Boolean 5
STOP

Obr. 1-1


Untitled 1 Block Diagram *
File Edit Operate Tools Browse Window Help
13pt Application Font
Thermometer
Time Stamp
Amplitude
Waveform Chart
output cluster
Boolean
Boolean 2
Boolean 3
Boolean 4
Boolean 5
stop

Obr. 1-2

1.2.3 Datové typy jazyka G

Stejně jako klasické programovací jazyky má i G datové typy, které charakterizují proměnné, konstanty, funkce, třídy a objekty. V LV jsou nositeli datových typů nejčastěji výše uvedené terminály a datové vodiče. Každý takový vodič může přenášet jen data určitého typu a může být připojen pouze k terminálu stejného typu. Každý datový typ je jinak barevně odlišen.

Programovací jazyk G má mnoho (odhadem 30) datových typů. Většina nume-rických datových typů jako integer, real atd. je známá z klasických programovacích jazyků, stejně tak string a boolean. Při programování se neobejdeme bez typu pole (array), který smí obsahovat jen položky jednoho definovaného typu. Novinkou pro mne byl datový typ **cluster**, který v sobě umožňuje sdružovat více datových typů, které mohou být naprosto odlišné. Dalším specifickým typem je **variant**, který se vyskytuje i u jiných programovacích jazyků, ale má jinou charakteristiku. **Variant** v LV nese informaci o **controlu** nebo **indicatoru**, dále informace o datovém typu i data samotná.

1.2.4 Proměnné a konstanty v prostředí LabVIEW

Co se týče proměnných není LV žádnou výjimkou a rozlišuje dva druhy, lokální a globální. Záleží, k jakému účelu ji budeme potřebovat, ale její vytvoření není nikterak náročné. Pouze v případě globální proměnné bych podotkl, že má větší režii a může běh programu zpomalovat.

Konstanty mohou být v podstatě jakéhokoliv typu, dokonce lze využít některých předdefinovaných, z numerických bych uvedl π, fyzikálních např. *e*. Při tvorbě aplikace jsem konstanty nejvíce použil při nastavování časových smyček a konstantní textové řetězce pro načítání URL (Uniform Resource Locator) adresy.

1.2.5 Řídící struktury

Do řídicích struktur jazyka G řadíme obecně známé programové cykly jako je **FOR** a **WHILE**, dále rozhodovací strukturu **CASE** a specialitou jazyka G je struktura **SEQUENCE** a **FORMULA NODE**.

Vesměs se jedná o smyčky a takovou mají i grafickou podobu. Vytvoření takové struktury je velice jednoduché, stačí si kliknutím v paletovém menu vybrat jakou z výše uvedených struktur chceme použít, tažením myši ohraničit terminály, které budou uvnitř dané struktury a pak jen definovat potřebné podmínky pro běh struktury.

Nejvíce využívanou strukturou byla při tvorbě aplikace byla smyčka **WHILE**, která ve spojení s terminálem **Wait until next multiple ms** tvoří časovou smyčku, vykonávající cykly s přednastaveným rozestupem, čímž se dá trochu ulehčit systému, který se mezitím může věnovat jiným úkonům.

1.2.4 Vizualizace

Vizualizaci v LV bych rozdělil na dvě oblasti. První se týká vizualizace datového toku, která pomáhá hlavně při tvorbě aplikací. Ve spojení s krokováním je to silný prostředek pro ladění aplikace, hlavně pokud hledáme, kam až aplikace běží podle očekávání a od jakého bodu generuje chybu. Při průchodu dat terminálem se u něho zobrazí okénko s aktuální informací, odesílanou z výstupu terminálu dále. Nevýhodou této vizualizace je podstatné zpomalení běhu aplikace, čímž se stává nepoužitelnou u procesů vyžadujících rychlý přenos a vyhodnocení dat.

Druhou oblast tvoří prostředky pro vizualizaci samotných dat. Podle typu dat, která chceme zobrazovat je můžeme rozdělit na binární, numerické a textové. Binární jsou různé LED, spínače a přepínače. Co se týče vizualizace numerických dat, nabízí LabVIEW ukazatele výšky hladiny, teploty, dále poměrně dobrý výběr grafů ve 2D a 3D provedení, záleží jen na tom, jaká data chceme zobrazovat. Při tvorbě aplikace jsem nejvíce využil **XY Graph** a **LED indicator** a **String indicator**.

1.3 Prostředky pro komunikaci

V současné době si lze již těžko představit jeden komplexní program, který by měl všechny možné funkce a vykonával námi požadované úkony. Spíše je kladen důraz na to, aby jednotlivé programy, které by spolu mohli a měli spolupracovat, byli schopné se vzájemně dorozumět. K tomu slouží komunikační rozhraní a protokoly. Verze 7.1 nabízí mnoho způsobů, jak komunikovat s okolními aplikacemi (TCP, UDP, IrDA, SMTP atd.). Mezi tyto prostředky patří také technologie **DataSocket** a **ActiveX**. Nyní si vyložíme pár základních pojmů a poté bude následovat podrobnější popis výše uvedených technologií.

COM – je to zkratka anglického Component Object Model,což je hlavní komponentová technologie OS Windows, umožňující komunikaci. Samotný COM je prázdný rámec, který poskytuje rozhraní objektů a dokáže zaručit jejich propojitelnost.

DCOM – Distributed COM, umožňuje připojení i ze vzdáleného počítače.

DLL – Dynamic Link Library. Je to soubor obsahující různé funkce, procedury, data atp., který je zaváděna do operační paměti za běhu programu.

DDE – Dynamic Data Exchange. Jedná se o komunikační protokol od firmy Microsoft, který umožňuje v rámci OS Windows programům vyměňovat si vzájemně data a instrukce.

OLE – Object Linking and Embedding. Umožňuje uživateli začleňovat data z jedné aplikace do jiné. Zachovává se vzájemná vazba, díky niž dochází k automatické aktualizaci objektu. Využívá DDE.

TCP/IP – Transmission Control Protocol/Internet Protocol. Je to sada internetových protokolů, pro komunikaci v počítačové síti.

API – Application Programming Interface. Jedná se o rozhraní pro programování aplikací, obsahující soubor procedur, funkcí či tříd libovolné knihovny . Určuje, jak do dané knihovny přistupovat.

1.3.1 DataSocket

Je to programovací technologie, založená na standardu TCP/IP, která zjednodušuje datovou výměnu mezi aplikacemi v rámci jednoho počítače, ale i počítačové sítě. Sjednocuje zavedené komunikační technologie měření a automatizace stejně, jako třeba webový prohlížeč sjednocuje odlišné internetové technologie v jeden lehce použitelný nástroj. DataSocket tak dělá z VI´s jednoduše konfigurovatelné a autonomní systémy. Byl vyvinut proto, že většina nástrojů užívaných v technologiích (TCP/IP, DDE) pro sdílení dat mezi aplikacemi nebyla schopna přenášet data v reálném čase (live data). Sestává ze dvou částí, DataSocket API a DataSocket server.

DataSocket API je nezávislý na protokolu, jazyce i operačním systému, navržený tak, aby co možná nejvíce zjednodušil interpretování binárních dat. Automaticky převádí uživatelem naměřená data na sled bytů, které jsou poté posílány dál a pak jsou data zpět převáděna do původní podoby.
Práce s DS API není složitá a skládá se hlavně ze základních akcí jako je otevření, čtení, zápis a zavření relace. V našem případě je využíváno pro čtení dat z OPC serveru.

DataSocket server je samostatná komponenta, pomocí které může program využívající DataSocket API rychle přenášet aktuálně měřená data přes internet souběžně několika vzdáleným klientským aplikacím.

1.3.2 ActiveX

Jedná se o soubor technologií firmy Microsoft, které dovolují uživateli spojovat jednotlivé programy a přizpůsobit je vlastním potřebám. Základ tvoří technologie COM a rozšiřuje původní technologii OLE.

ActiveX je událostmi řízená technologie, což znamená, že nastane-li nějaká událost, jsou dané programy informovány, aby uživatel či program mohl na nastalou událost adekvátně reagovat.

Popisovaná technologie se dělí na oblast ActiveX automatizace a ActiveX řídicí prvky a zásobníky. První odkazuje řídicí proces jednoho programu na program jiný právě skrz ActiveX. Podobně jako v počítačových sítích, jedna aplikace vystupuje jako klient a další jako server. LV podporuje vzájemnou nezávislost aniž by programy ztratily schopnost sdílení informací navzájem. Toto sdílení je ukládáno během komunikace klienta s ActiveX objekty, určenými serverem. Objekty mají, jak je známo, vlastnosti a metody a k těm může být přistupováno přes klienta. Vlastnosti jsou jednoduše znaky objektu, jejichž parametry mohou být nastavovány či měněny z ostatních programů. Metody jsou funkce vykonávané objektem a ty mohou být iniciovány ostatními programy.

Nejběžnější použití ActiveX je přes ovládací prvky, což jsou komponenty, existující uvnitř zásobníků. Jakýkoliv program fungující jako Active-X zásobník dovoluje uživateli do něj ovládací prvky vkládat. Ovládací prvky pak čerpají svou funkčnost a vlastnosti z těchto zásobníků. Takový zásobník je například LabVIEW, stejně tak jako může být klientem či serverem.

2 OPC

Zkratka OPC znamená OLE for Process Control, neboli volně přeloženo, spojování a vkládání objektů do řízení procesů. Představuje úspěšný pokus o standardizaci komunikačních rozhraní v oblasti průmyslové automatizace. Ta se skládá ze dvou skupin, v první jsou průmyslové automaty, akční členy a čidla. Do druhé patří řídicí a operátorské počítače a průmyslové informační systémy. Výhoda této standardizace spočívá v tom, že uživatelé již nejsou nuceni odebírat jak hardware, tak software výhradně od výrobců, jejichž produkty podporují pouze zavedené protokoly. Standard OPC spravuje nezisková organizace OPC Foundation [2]. Ta sdružuje nejvýznamnější světové výrobce, zabývající se monitoringem, vizualizací a ostatními aplikacemi z odvětví řízení a sledování technologických procesů. Implementace standardu jsou dostupná zdarma a všem.

2.1 Princip OPC

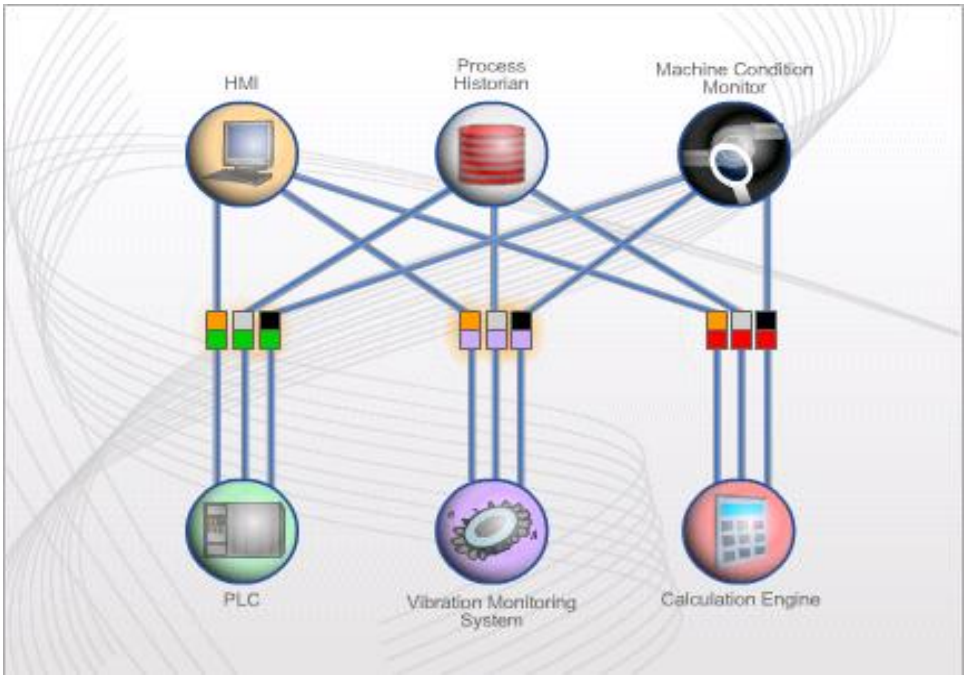
Dříve každá aplikace musela obsahovat ovladač k danému zařízení, vznikaly konflikty mezi ovladači od různých výrobců, kdy více než jeden ovladač nemohl přistupovat k požadovanému hardwaru najednou (viz Obr. 2-1). Tyto problémy odstraňuje popisovaný standard.

OPC vytváří aplikační vrstvu mezi výše zmíněnou první skupinou průmyslové automatizace a programy, které s prvky této skupiny komunikují.

Počítače opatřené OPC serverem, který nemusí být nutně od výrobce ovládaného zařízení, jsou dnes již schopny celkem jednoduše a bezproblémově komunikovat s daným zařízením. To vše za předpokladu, že je nainstalován náležitý ovladač (driver) a je k dispozici klient, schopný se serverem komunikovat.

Základ standardu tvoří komponentová technologie COM. Z názvu standardu vyplývá, že OPC je nadstavba k technologii OLE, i když dnes se rozdíl mezi COM a OLE stírá. Firma Microsoft používá COM k implementacím velkého množství komponent v OS Windows, tudíž je model široce podporován. Obsahuje cenné vlastnosti pro OPC, jako příklad

bych uvedl globální registraci komponent, která umožňuje klientovi snadno zjistit, jak se má k OPC serveru dostat. Dále to jsou kategorie komponent, díky nimž můžeme snadno zjistit, jaké OPC servery jsou na daném počítači nainstalovány a nemusíme, mnohdy zdlouhavě, hledat v paměti PC.



Obr. 2-1

2.1.1 OPC Specifikace

OPC standard rozlišuje několik specifikací, které určují pravidla nastavení a chování rozhraní. Specifikací je celkem deset, záleží na tom, kde a jak chceme standard používat. Pro přehled budou uvedeny jen ty nejpoužívanější.

OPC Data Access (DA) – je nejčastěji využívanou specifikací. Je navržena jako prostředek usnadňující uživatelům přístup k zařízením, resp. jejich datům.

OPC Alarms and Events – neboli signalizace a události. Udává jak podle nastavení obsluhovat zaznamenané události, které zahrnují výstrahy, informační zprávy, akce obsluhy a zprávy sledování a revize.

OPC Historical DA – jak již název napovídá, popisuje tato specifikace přístup k dříve uloženým datům.

OPC XML DA – umožňuje implementovat specifikaci na libovolný počítač, bez ohledu na jeho operační systém, za předpokladu, že podporuje HTML a XML. Využívá technologie DCOM, což umožňuje přistupovat k OPC serveru po internetu z jakéhokoliv místa na světě.

2.1.2 OPC Server

Jedná se o program, který uživateli poskytuje možnost komunikace se zvoleným zařízením průmyslové automatizace, pokud server má nainstalován příslušný ovladač k zařízení. V závislosti na tom, co s daty chceme dělat (vizualizace, archivace...), je zvolen takový server, který odpovídá určité specifikaci standardu OPC. V dnešní době je u většiny světových výrobců zařízení dobrým zvykem dodávat ke každému výrobku či řadě vlastní OPC server. Jelikož některá starší zařízení vlastní OPC server nemají, není někdy lehké ho pro dané zařízení sehnat. Existují firmy, které ale tyto servery vyvíjejí a při dostatečném úsilí při vyhledání na webu je lze objevit a stáhnout si demoverzi serveru, abychom ho mohli odzkoušet.

2.1.3 OPC klient

Klient je v podstatě HMI, neboli Human Machine Interface, což je aplikace, která umožňuje uživateli kontrolovat a ovládat průběh vybraného procesu . Jediným požadavkem pro úspěšnou komunikaci dat mezi serverem a klientem je jejich shodná specifikace standardu OPC. Většina OPC serverů se spouští automaticky ihned po inicializaci klienta.

Data předávaná serverem klientovi tvoří tři složky:

- Hodnota např. měřené veličiny, časová známka (timestamp) a kvalita. Hodnoty veličin mohou mít různý datový typ (numerický, booleovský, textový).
- Časová známka udává přesný čas, kdy byla hodnota naměřena.

- Kvalita vypovídá o tom, zda je přenos dat mezi zařízením a serverem v pořádku, nebo došlo k nějaké chybě apod. Rozlišujeme tři základní kvality, good (dobrá), bad (špatná) a uncertained (neurčitá).

2.2 Použitý OPC server

Ačkoliv je standardizace poměrně pokročilá, jak již bylo výše řečeno, není možné použít libovolný OPC server ke komunikaci s vybraným zařízením. Nejprve jsem sháněl jakýkoli OPC server, resp. jeho demoverzi, která by byla volně ke stažení. Prvním uživatelsky srozumitelným a přehledným serverem byl Matricon OPC Server for Simulation and Testing. Na webových stránkách firmy Matricon[3] je po bezplatné registraci možné si stáhnout kromě demoverzí několika OPC serverů různých specifikací i drivery k zařízením od několika výrobců a webová vysílání, která mají pomoci při řešení problémů k danému tématu. Pokud je uživatel začátečníkem v oblasti OPC a zvládá angličtinu alespoň na průměrné úrovni, může absolvovat úvod do problematiky díky několika instruktážním videím.

Jak název uvedeného serveru napovídá, je určen k simulaci a testování. Tudíž mi celkem dobře posloužil otestování vizualizační aplikace vytvořené v LabVIEW, jejíž tvorba bude vyložena v samostatném oddíle.

Vyhledání správného serveru, který by obsahoval vhodný driver pro daného řídicího systému zabralo pár hodin, ale nakonec bylo mé hledání úspěšné. Shodou okolností liberecká firma Kontron Czech s.r.o. nabízí na svých webových stránkách[4] vhodný OPC server. Po registraci jsem si mohl stáhnout potřebný server, vhodný pro zadaný řídicí systém firmy Mitsubishi Electric, který je produktem firmy Kepware Technologie a má označení KEPServerEx. Specifikace serveru je Data Access.

2.2.1 Instalace Serveru

Vlastní instalace serveru je celkem bezproblémová, pokud máme podporovaný operační systém a dostatek místa na pevném disku. Uživateli je nabídnuto, jaké drivery chce nainstalovat, podle toho s jakými zařízeními

má server komunikovat. Mimo jiné je tu i možnost instalace OPC Quick Client, který se využívá při testovacím provozu. Po úspěšném instalování serveru je třeba nastavit dobře DCOM, aby správně probíhala komunikace mezi serverem a lokálními či vzdálenými klienty.

Spustit nastavení DCOM můžeme buď přes příkazový řádek nebo přes funkci implementovanou přímo do prostředí serveru, která se nachází na nástrojové liště. Podle toho, jestli k serveru mají přistupovat pouze lokální nebo vzdálení klienti musíme nastavit výchozí vlastnosti a zabezpečení. U některých operačním systémů se doporučuje ještě správně nastavit firewall.

2.2.2 KEPServerEx

Pro přístup ke každému zařízení slouží samostatný komunikační kanál (channel),což je jedna ze základní komponent serveru. Další komponentami jsou zařízení (device), Tag Group a Tag. U klientů se značí „tagy“ jako item, který reprezentuje jednu hodnotu sledované veličiny. Každý komunikační kanál může obsahovat několik zařízení, ty mohou mít libovolně skupin, podskupin a ty zase potřebný počet „tagů“. Tag bych přirovnal asi nejlépe k čidlu v měřicím řetězci.

Při zakládání nového komunikačního kanálu je potřeba vybrat podle použitého zařízení správný driver, dále komunikační parametry a metodu optimalizace hodnot. Mezi parametry komunikace patří ID, neboli unikátní identifikátor COM, který může mít hodnotu 1 až 100. Dále přenosová rychlost, počet datových a stop bitů, parita atd.

Při založení nového zařízení se nastavuje jméno, model vybraného zařízení, následuje konfigurace komunikačních časovacích parametrů jako je timeout připojení, žádosti, hlášení chyby po určitém počtu timeoutů. Dále se nabízí možnost přerušení komunikace se zařízením na určitou dobu, jestliže nastane porucha komunikace. Dá se tak vyhnout možným problémům v komunikaci s jinými zařízeními na stejném kanále.

Poslední co je třeba založit je Tag. Ten můžeme založit buď staticky v rozhraní serveru nebo dynamicky pomocí OPC klienta. Statické založení zahrnuje pojmenování veličiny, určení adresy v paměti zařízení, datový typ,

klientský přístup (pouze ke čtení, čtení/zápis) a rychlost snímání. Narozdíl od dynamického založení má statické několik výhod jako je např. prohledávání veličin v klientech podporujících tuto funkci a škálování (žádné, lineární, kvadratické). Pokud zadáváme více než jeden Tag, je možné zadat pouze počáteční a další se už vygeneruje automaticky pomocí volby duplicate.

Popisovaný server podporuje všechny doposud vydané verze specifikace Data Access (poslední je verze 3). Verze 2 mino jiné ustanovuje čtyři základní způsoby komunikace klient/server. Jsou to následující:

Synchronní komunikace se zařízením – dokončuje požadavek klienta pro čtení nebo zápis až po té, co jsou data přenesena od nebo do zařízení.

Synchronní komunikace s cache – data jsou nejdříve načteny do vyrovnávací paměti. Odtud jsou zasílány klientům. Periodická obnova dat je záležitostí serveru a není závislá na komunikaci s klienty.

Asynchronní komunikace – klient zasílá pouze požadavky na čtení či zápis dat a po komunikaci serveru se zařízením je klientovi podána zpráva.

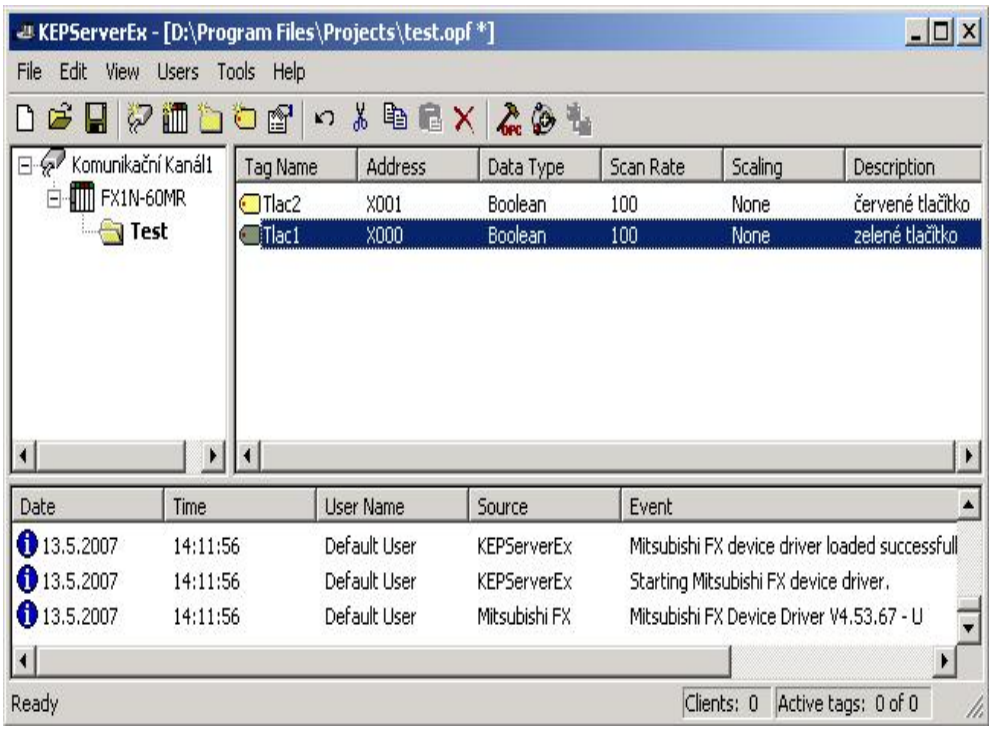
Periodická komunikace – klient je informován pouze při změně požadovaných hodnot, které jsou periodicky načítány a uchovávány v cache.

Způsob komunikace se nastavuje u každé veličiny zvlášť v OPC klientovi, takže je možné způsoby komunikace různě kombinovat.

OPC servery většinou zobrazují svá data jako samostatné položky, ale existují i takové servery, které umožňují zobrazování v podobě polí. Tato vlastnost umožňuje přenášet celé datové bloky, což se může zdát jako velice výhodné a v případě, že pole nejsou příliš veliká tomu tak i je. Ale v případě větších datových struktur vyvstává problém s tím, že alokace takových celků zvyšuje nároky na režii a zpomaluje tak běh celého systému.

Potřebujeme-li přenášet pouze malou oblast paměti, třeba samotnou hodnotu, je použití polí také spíše na škodu než k užitku, jelikož je nutné přenést celé pole hodnot. Záleží na driveru, jak velké bloky paměti umožní přenášet. V našem případě je přenášení polí hodnot sice možné, ale driver dovoluje načítat pouze celé datové oblasti, což znamená přenášet najednou minimálně 256hodnot a v případě, že není celý adresový prostor využit, se mi jeví taková možnost celkem zbytečná.

Pro ilustraci je na následujícím obrázku uvedeno rozhraní serveru.



Obr. 2-2

3 Tvorba aplikace

V této části práce je popsána tvorba vizualizační aplikace ve vývojovém prostředí LabVIEW. Požadavky na aplikaci jsou následující:

- Schopnost komunikace s OPC serverem.
- Přehledná a srozumitelná vizualizace dat.

Při vývoji byli využity více či méně všechny nástroje pro vizualizaci komunikaci, uvedené v první části.

3.1 Komunikace

Navázání komunikace s OPC serverem je realizováno přes subVI Data Socket OtevriSpojeni. Uvnitř něj byli, pro přehlednost, vytvořeny dva sub VI´s, první zajišťuje tvorbu referencí a druhý pro navázání spojení s URL, který jednoznačně určuje umístění komunikovaných dat v adresovém prostoru OPC serveru. Po navázání spojení je třeba data načítat, což má na starosti subVI DataSocket ObnovData.

O zakončení komunikace je postaráno terminálem DataSocket Zavri Spojeni, který má stejně jako terminál s opačnou funkcí dva subVI´s. Ty zařídí odpojení aplikace od zdroje dat a uvolnění referencí z paměti.

3.1.1 DataSocket VytvorRef

Jak je již patrné z názvu, vytváří tento terminál reference na nějaký objekt. Hlavními prvky jsou funkce ActiveX, Automation Open a Create ActiveX Event Queue. První vrací automaticky referenční číslo (refnum), které odkazuje na ActiveX objekt třídy CWDSLib. Druhá zmíněná funkce vytváří frontu událostí a je vytvořena pro události OnStatusUpdated a OnDataUpdated. Výstupem popiso-vaného terminálu je vedle chybového kanálu také cluster, jehož vstupy tvoří výstup z první popsané funkce, obou událostních front a z uzlu vlastnosti Data, specifikující aktuální hod-notu a znaky, které reference přijala od zdroje dat.

3.1.2 DataSocket Pripoj

Tento terminál spojuje dodané reference s určeným URL. To zajišťuje uzel volající metodu ConnectTo, jež přivedenou referenci spojí s nadefinovanou URL. Metoda kromě zdroje dat požaduje nastavit také přístupový mód. Pro automatické načítání dat ze zdroje při jakékoliv jejich změně byl
zvolen mód cwdsReadAutoUpdate. Případné chyby zjišťovány terminálem DataSocket Set Error From Status.

3.1.3 DataSocket ObnovData

Z clusteru vstupní reference je separováno přes funkční blok Unbundle refnum odkazující na již výše zmíněnou třídu CWDSLib. Na tu je aplikována metoda Update, která zajišťuje načtení dat. Výstup z metody a vstupní reference jsou přivedeny do terminálu NastavError.

3.1.4 DataSocket Odpoj

Jako u předchozího terminálu je separováno stejné refnum, na nějž je užita metoda Disconnect, která provede odpojení a výstup je sveden do NastavError.

3.1.5 DataSocket UvolniRef

Vstupní reference jsou vzájemně separovány a fronty událostí ActiveX jsou zrušeny a refnum odkazující na ActiveX objekty ukončeny. Výstupem je pouze kanál chybových hlášení.

3.2 Převod dat

Jelikož data z OPC serveru jsou typu Variant a vstupy do vizualizačních terminálů mohou vstupovat pouze numerické nebo textové datové typy, je třeba tato data převést do použitelné podoby.

Tuto funkci realizuje terminál DataSocket CtiDataInt32 sestávající ze subVI DataSocket CtiVariant a struktury Case. Struktura má dva stavy, v nichž je buď příchozí hodnota nulována nebo pomocí funkčního bloku

Variant to Data převést na typ Int32, pro LabVIEW použitelný. Pomocí tohoto terminálu je také získávána informace o kvalitě přenosu, podle standadru OPC.

Změnou typu dat lokální proměnné, přivedené do funkce Variant to Data lze dosáhnout, toho aby výstupem byla třeba data typu Boolean.

Dále je třeba získat z přivedených dat informaci o čase (timestamp), která je zobrazována u dané veličiny a také je použita při vykreslování závislosti veličiny na čase. K tomu byl vytvořen terminál DataSocket OPCCas.

K tomu, aby byla data přehledně zobrazována je musíme po určitou dobu uchovávat, a postupně je vykreslovat do grafu, jinak totiž nevidíme nějaký rozumný průběh, ale většinou jen bod nebo přímku. K dočasnému uchovávání polí hodnot byl vytvořen buffer, ze kterého jsou data přivá-děna na XY Graph. V případě, že jsou zobrazovány pouze data ve formátu boolean, je buffer zbytečný.

3.2.1 DataSocket CtiVariant

Hlavní strukturou tohoto subVI je Stacked Sequence, která v podstatě umožňuje provádět příkazy postupně po okénkách, v nichž jsou umístěny.
Z venku do struktury vstupují reference, časovač, chybové hlášení a prázdný atribut. V každé fázi struktury se nachází rozhodovací struktura Case, která na základě splnění podmínky provede požadovanou funkci.

V první fázi je určující hodnota časovače, kdy program buď čeká na dostupná data, nebo na data nečeká a posílá aktuální data či vrací hodnotu false.

Pro ověření dostupnosti dat byl vytvořen terminál DataSocket Dostup Data, který za pomoci metody DataUpdated zjišťuje, zda od posledního čtení byla data změněna.

V druhé fázi je zjišťováno pomocí metod HasAttribute a GetAttribute třídy ICWData, zda se ve vstupní referenci nachází atribut, pokud ano, tak je z něho následně získána hodnota a reference je ukončena. Jestliže je atribut prázdný, je hodnota získána ze vstupní reference ihned.

Nastane-li chybový stav, je ve třetí fázi odbaven. Pokud nenastane, program pokračuje beze změn dál.

3.2.2 DataSocket OPCCas

Jak již bylo řečeno, tento terminál načítá ze vstupní reference informaci o čase, ale tu je třeba převést z času OPC serveru na čas používaný Lab VIEW, což je realizováno terminále PrevedOPCCas, do kterého vstupují načtené hodnoty ze dvou terminálů CtiDataInt32. Výstupem terminálu je čas v sekundách. Ten je následně převáděn funkcí Formate Date/Time String na datum a čas v obecně uznávaném formátu den:měsíc:rok, hodiny:minuty:sekundy.

3.2.3 Buffer

Umožňuje pracovat jak s jednorozměrným polem, tak s vícerozměrným. Operacemi se složkami pole je postupně dosaženo toho, že jsou data postupně vykreslována a je-li buffer naplněn, jsou stará data postupně nahrazována novými.

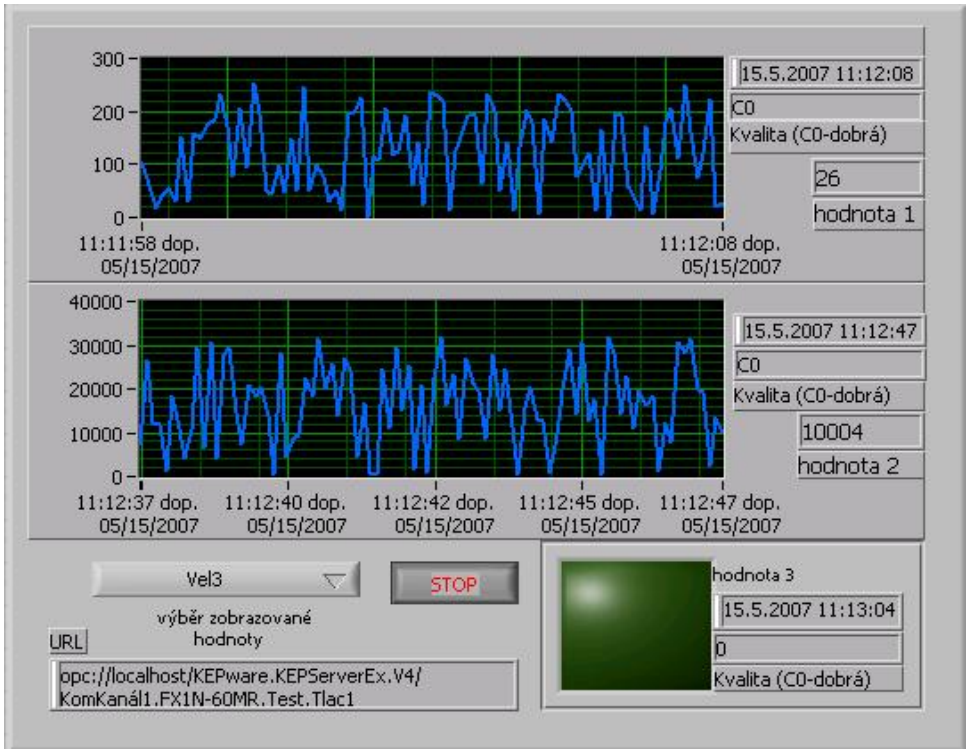
3.3 Hlavní program

Při spuštění programu je spojení s OPC serverem neaktivní. na výběr jsou čtyři možnosti, nezobrazovat žádnou veličinu (volba nic) nebo zobrazovat jednu ze tří nabízených (Vel1,Vel2,Vel3).K zobrazení prvních dvou je použit XY Graph, třetí je dvoustavová, tudíž k indikaci postačí LED. U každé zobrazované veličiny je uváděna její aktuální hodnota a časová známka (timestamp). Ve spodní části panelu je zobrazována URL adresa vybrané veličiny. K ukončení chodu programu slouží tlačítko STOP. Pro názornost je uveden Obr. 3-1.

Program funguje následujícím způsobem. Uživatel si vybere, jakou veličinu chce zobrazovat, díky implementované funkci DataSocket Select si může vybrat ze všech OPC serverů, nainstalovaných na daném počítači. Jedná se o tzv. OPC browsing, kdy je uživateli umožněno prohledávat adresový prostor serveru a vybrat si přesně jakou veličinu chce z OPC

serveru komunikovat.Cesta k veličině je ve tvaru opc://jméno_počítače /jméno_serveru/komunukační kanál.zařízení.skupina.veličina. V našem případě je jméno počítače localhost, jelikož je server nainstalován na stejném počítači jako klient.

Každá volba má vlastní smyčku ve struktuře CASE. Tato struktura je navíc vnořena do smyčky WHILE. Po výběru veličiny je URL předána do příslušného okna struktury, kde je přivedena do terminálu DataSocket OtevriSpojeni, který naváže komunikaci se serverem.

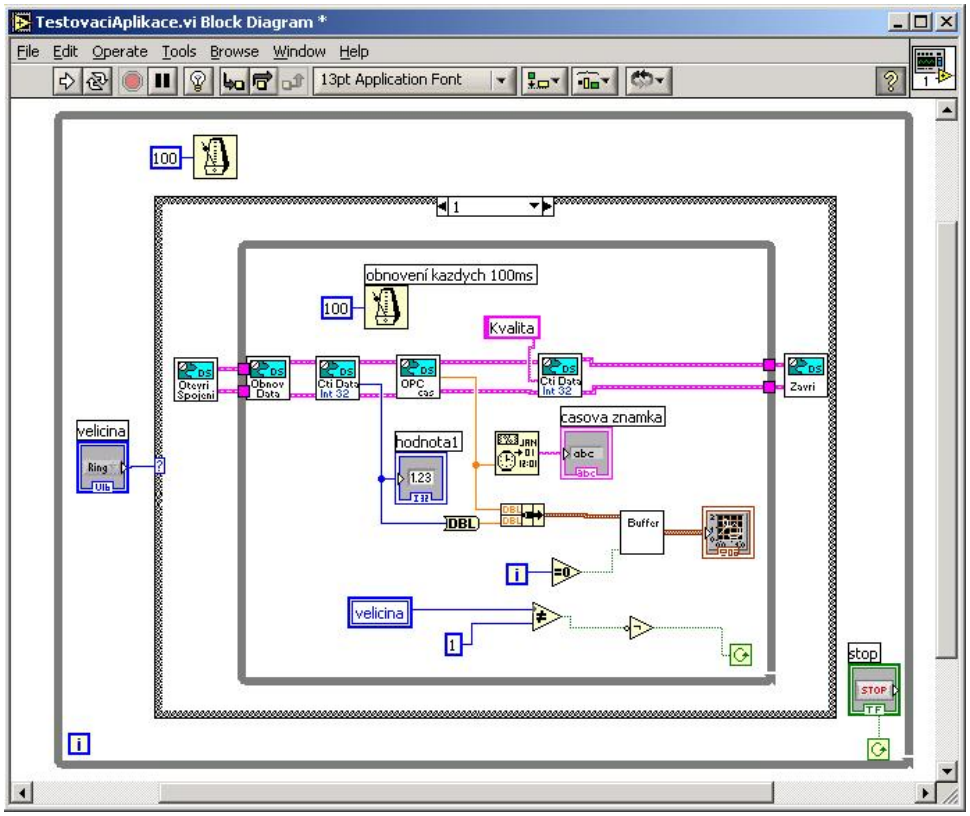


Obr. 3-1

S daty je dále pracováno v další smyčce WHILE, vnořené v každém okně smyčky CASE. Zde jsou data načítána terminálem DataSocket ObnovData a následně z nich jsou získávány potřebné složky, jako je hodnota, kvalita a čas, které jsou zobrazovány na hlavním panelu aplikace.

Jestliže si uživatel vybere jinou veličinu, je komunikace s dosud používanou veličinou, případně i celým serverem ukončena a uživatel je opět nabídnuto vybrat si mezi přítomnými OPC servery.

K vytvoření celé aplikace přispěla nejen kvalitně a celkem podrobně vypracovaná nápověda vývojového prostředí LabView, ale i fórum na webových stránkách jeho výrobce [1]. Pro představu je uveden blokový diagram hlavního programu (viz Obr. 3-2). Za téměř každým terminálem, vyjma originálních, je třeba si představit další blokové digramy.



Obr. 3-2

4 Rozhraní průmyslové komunikace

Jak již název práce napovídá, při jejím řešení bylo využito řídicího systému, konkrétně programovatelného automatu MELSEC PLC, řady F1 XN, typ 60-MR. Toto PLC má díky přídavným modulům možnost komunikace přes standardní komunikační rozhraní RS232, RS485/RS422, dále pak Profibus/DP, Device Net, CANopen, AS-interface, CC-Link.

Driver, se kterým bude vytvořená aplikace porovnávána, byl naprogramován pro komunikaci po RS-232, proto bude v této kapitole vyložen princip tohoto rozhraní a také nastíněna funkce daného driveru.

4.1 Rozhraní RS232

Toto komunikační rozhraní je též nazýváno sériová linka a konektor na zařízení je sériový port. V oblasti osobních počítačů se od sériového portu upouští a je nahrazován jiným druhem sériového komunikačního rozhraní a tím je USB (Universal Serial Bus). V průmyslových aplikacích, hlavně tam, kde je třeba vysokých přenosových rychlostí se již USB také prosazuje, ale RS232 a jeho modifikace RS485/RS422 mají zatím nezastupitelnou roli. Nevýhodami jsou relativně nízká přenosová rychlost a odolnost proti vnějšímu rušení.

4.1.1 Fyzické parametry

Z hlediska signálů používá RS232 dvě napěťové úrovně. Pro logickou 0 to je +3 až +15 V a pro logickou 1 rozmezí -3 až -15 V. Odpor zátěže se pohybuje mezi 3 a 7 Ω a její kapacita by neměla přesahovat 2,5 μF. Dle standardu by délka vodiče měla být maximálně 15m, což lze při dodržení podmínek na kvalitu vodiče prodloužit, snižuje se tak ale přenosová rychlost, která je maximálně 19200 Bd.

4.1.2 Přenosové parametry

Základní pojmy pro přenos jsou velikost rámce, bity a velikost přenosové rychlosti.

Character Frame – znakový rámec, skládá se z řídící a datových bitů. Přenášený znak je zakódován v datových bitech (Data Bits).

Start Bit – je jím uvozen počátek rámce. Klidový stav linky je charakterizován log. 1. Start Bit má tedy hodnotu log. 0.

Data Bit – udává počet datových bitů, které rámec obsahuje. Je jich 5 až 8, podle potřeby a následují v obráceném pořadí ihned po start bitu. Obráceným pořadím je míněno, že nejprve je přenášen nejméně významný bit (LSB) a jako poslední datový bit je nejvíce významný bit (MSB).

Parity Bit – patří mezi řídicí bity, většinou následuje za datovými bity. Slouží ke kontrole správnosti přenosu, kdy přijímač a vysílač mají určeno, zda je parita lichá, či sudá a v případě, že kontrolní součet liší od dohodnuté konvence, přijímač ví, že nastala chyba přenosu.

Stop Bit(s) – může být jeden, jeden a půl, či dva stop bity, patří mezi řídicí. Jejich reprezentace je opačná než u start bitu a uvozuje konec rámce.

Baud Rate – udává přenosovou rychlost dat mezi zařízeními. Odpovídá počtu změn za 1 sekundu. Většinou je ztotožněna s rychlostí v bit/s, ale neplatí to vždy. Obvyklou hodnotou je 9600 Bd.

4.1.3 Komunikace se zařízením

Výstupem z PLC je port pro RS422, z něhož je vyveden datový kabel. Ten je spojen převodníkem RS422/RS232 a připojen na sériový port počítače. Vlastní komunikace je pak řízena ovladačem.

4.2 Driver RS232

Porovnávaný driver byl naprogramován v prostředí LabVIEW, ale jelikož je majetkem firmy Preciosa a.s., nemohu zde uvést jeho přesnou strukturu, ale pokusím se přiblížit jeho funkci a vlastnosti.

4.2.1 Vlastnosti driveru

Ovladač umožňuje v uživatelském rozhraní nastavit komunikaci celkem ve čtyřech režimech a to čtení, zápis, ForceON a ForceOFF. Potom lze nastavit čas vypršení (timeout), po jehož vypršení je hlášena chyba. Ta nastane v případě, že driver na svůj dotaz neobdrží odpověď.

V režimu čtení může uživatel nastavit délku čtených dat a samozřejmě jejich adresu. Hodnota čtených dat je pak zobrazována v textovém okně.

Ze stejného místa jako pro čtení je adresa zadávána i v režimu zápisu. Data jsou zapisována v podobě jednorozměrného pole hodnot.

Pro přehlednost a ověření je zasílaný dotaz driveru zobrazován v samostatném dialogovém okně.

4.2.2 Princip driveru

Pro to, aby vůbec komunikace mohla proběhnout je nejprve potřeba inicializovat sériový port. To obstarává k tomu navržená utilita. Nastavuje také komunikační parametry sériového portu. Jejími vstupními parametry jsou baud rate, velikost a bufferu, počet datových a řídicích bitů. Tyto informace jsou pak pomocí globální proměnné předávány do programu driveru. K vytvoření takovéto utility jsou používány funkce z knihovny VISA prostředí LabVIEW.

Program driveru funguje následujícím způsobem, na základě volby je určen identifikační prefixní řetězec znaků, ten je zkoncentrován s řetězcem dotazu, a ukončovacím řetězcem. Řetězec dotazu udává adresu a délku dat v případě čtení, či vstupní pole při zápisu. Délka dat a vstupní pole jsou nejprve z číselné podoby převedeno na string.

Po té, co je proveden kontrolní součet jsou data předána sériovému portu.

Po obdržení odpovědi od zařízení je chod programu driveru ukončen.

4.2.3 Doporučení

Spouštění driveru ve více oknech najednou se nedoporučuje z důvodu možného konfliktu. Dále nedoporučuji paralelní běh OPC serveru a driveru.

Server totiž používá svůj vlastní driver pro komunikaci s řídicím systém a v případě, že je již spuštěn jiný driver, který se systémem komunikuje přes stejný port, tudíž může fungovat komunikace jen s jedním programem.

V případě, že byl nejprve používán driver vytvořený v LabVIEW a chceme spustit aplikaci komunikující se zařízením přes OPC server, je nutné LV restartovat. V opačném případě je nutné server vypnout nebo alespoň deaktivovat komunikační kanál využívající implementovaného driveru a spustit inicializační utilitu pro driver naprogramovaný v LV.

5 Porovnání OPC serveru a driveru

Vlastnosti použitého OPC serveru je třeba porovnat v kontrastu s poskytnu-tým driverem v několika oblastech. Vedle přenosových vlastností obou řešení je třeba také posoudit uživatelské aspekty.

5.1 Přenosové vlastnosti

Z přenosových vlastností bych jako hlavní označil rychlost přenosu dat ze zařízení do počítače, resp. do programu. Další přenosovou vlastností, která je hodna pozornosti je velikost adresovatelného prostoru paměti řídicího systému.

5.1.1 Rychlost přenosu

Měření přenosové rychlosti driveru jsem provedl ve dvou režimech. Pro tuto činnost byl použit jednoduchý program vytvořený v prostředí Lab VIEW.

Princip programu je následující, před spuštěním byl na driveru nastaven jeho režim (čtení/zápis) a dva parametry, adresa (která je pro měření nepodstatná) a délka čtených dat resp. velikost vstupního pole, kde n značí počet znaků. V momentě spuštění byl aktivován jeden časovač, který změřil počáteční čas. Když na zadaný dotaz přišla od řídicího systému odpověď, druhý časovač zaznamenal čas ukončení běhu programu. Čas počátku byl odečten a od času ukončení programu, výsledkem byl tedy rozdíl časů Δt udaný v milisekundách. Pro každou nastavenou délku dat bylo měření provedeno desetkrát, a následně byl aplikován aritmetický průměr.

Tab. 1 – Tabulka závislosti času přenosu na délce čtených dat

Délka čtených dat [n]	Δt [ms]										průměr [ms]
1	36	39	36	38	38	37	38	36	30	48	37,6
5	50	47	44	49	44	48	49	49	49	47	47,6
10	50	52	52	53	52	52	53	53	52	53	52,2
20	77	80	79	82	79	80	80	80	78	75	79
30	95	94	94	94	93	92	92	94	93	94	93,5

Tab. 2 – Tabulka závislosti času přenosu na velikosti vstupního pole

Velikost vstupního pole [n]	Δt [ms]										průměr [ms]
1	33	35	37	38	36	37	39	36	37	38	36,6
5	49	49	45	43	46	44	44	42	47	43	45,2
10	54	57	57	53	53	55	68	53	54	53	55,7
20	80	87	89	92	86	87	79	80	87	80	84,7

Z průměrných hodnot je patrné, že rychlost čtení a zápisu je téměř shodná, vzájemná odchylka činí maximálně 7,2%.

Z důvodu, že OPC server využívá driver ke stejnému zařízení, jako je výše uvedený, programovaný v LabVIEW, se lze domnívat, že přenosové časy obou technik se znatelně neliší.

V případě vizualizace je však potřeba počítat s tím, že data z OPC serveru je třeba dále zpracovat, převést do vhodného formátu a pak teprve zobrazovat. Měření času přenosu ze server do aplikace jsem provedl stejným způsobem jako u driveru. Časy jsou samozřejmě kratší, než při komunikaci se zařízením.

Tab.3 – Tabulka závislosti času přenosu na počtu „tagů“

počet adresovaný hodnot [n]	Δt [ms]										průměr [ms]
1	7	8	6	8	7	7	9	8	6	7	7,3
5	11	10	10	11	11	12	12	12	13	12	11,4
10	15	22	25	16	17	16	24	18	24	23	20

5.1.2 Velikost adresovatelného prostoru

Při popisu OPC serveru jsem se zmínil o možnostech adresace datových oblastí řídicího systému. Potřebuje-li uživatel dostat od řídicího systému souvislý blok dat, např. 50 adres, je vhodnější driver. Stačí nastavit jen počáteční adresu a délku čtených dat. V OPC serveru by toto znamenalo založit postupně 50 tagů. Naopak schopnost server komunikovat jednotlivé hodnoty je užitečné v případě, že potřebujeme jenom několik hodnot, z nesousedících adres. Porovnávaný driver není schopen adresovat najednou data různě rozmístěná v paměti řídicího systému.

5.2 Ostatní vlastnosti

Do této skupiny řadím uživatelské aspekty jako je přehlednost, srozumitelnost a snadnost či obtížnost ovládání, možnosti použití pro jiné řídicí systémy apod.

5.2.1 Přehlednost

OPC server je koncipován hierarchicky, což je možné vidět na obrázku Obr. 2-2. Myslím že z hlediska přehlednosti mu není co vytknout a orientace v něm je spíše intuitivní, než aby uživatel složitě hledal požadovanou položku. U porovnávaného driveru jde spíše o jeho funkčnost než nějakou estetickou stránku, ale v podstatě je také přehledný.

5.2.2 Srozumitelnost

Z hlediska srozumitelnosti výstupu je OPC server, resp. jeho klient daleko před driverem, jelikož ukazuje přímo měřené hodnoty, kdežto driver, jak již bylo zmíněno v jeho popisu , je vybaven dialogovým oknem, které zobrazuje přenesená data, ale ty jsou v "syrovém" stavu, pro uživatele neznalého architektury daného řídicího systému a driveru, jsou takřka nic neříkající. Pro případné další zobrazení např. v grafu by musela být tato data dále upravena.

5.2.3 Použitelnost

Použitelnost OPC serveru závisí na instalovaných driverech a jelikož při instalaci výše zmíněný server nabízel drivery k několik zařízením od celé řady světových výrobců, má tudíž velkou možnost uplatnění. Hlavní nevýhoda posuzovaného driveru spočívá v nemožnosti vzdáleného přístupu k zařízení. V případě OPC serveru je to záležitost nastavení přístupových parametrů a uživatelských práv k serveru, což je otázka maximálně několika desítek minut.

Z hlediska dynamického zpracování dat, obsluhování nějakého procesu a reakce na vzniklé situace, například v jednom okamžiku se zvýší v určitém místě teplota a na tuto událost je třeba zareagovat a o odeznění

zase reagovat na jinou nastalou situaci, je výhodnější porovnávaný driver. Ten je totiž na základě vlastního vyhodnocení může přistupovat k aktuálně nutným adresám narozdíl od OPC serveru, kde chceme-li k nějakému prostoru v datové oblasti PLC přistupovat, musíme nejprve navolit přístupové parametry a adresu

Závěr

Účelem práce bylo zjistit, zda je vhodné použít OPC technologie pro vizualizaci dat, dodávané řídicím systémem. To se díky použití grafického programovacího prostředí LabVIEW podařilo dokázat na příkladě. Vyvinutá aplikace byla koncipována jako testovací, tudíž k jejímu plnému využití v praxi by bylo třeba ji náležitě doladit a zkompilovat do přenositelného formátu.

Součástí práce také bylo srovnání řešení komunikace s řídicím systémem pomocí OPC serveru a driveru. Byli diskutovány klady a zápory obou řešení. V podstatě záleží na funkci, jakou by řídicí systém měl plnit. Pokud by měl být pouze prostředkem pro získávání dat, která by následně měla být zpracovávána do grafické podoby nebo by měla být dále uchovávána apod., je výhodnější použít OPC server. Osobně si myslím, že s rostoucí standardizací v průmyslové automatizaci a hlavně v oblasti komunikace informačních a řídicích systémů má OPC technologie veliký význam a investovat do ní má význam hlavně v případech, kdy je třeba sledovat a řídit rozsáhlejší systémy, čítající minimálně několik zařízení. Rozhodně by se nevyplatila v případě jednoho PLC. Výhodné je OPC technologii také využít při řízení a monitorování vzdálených provozů, kde je třeba zaručit jen spolehlivý přenos dat buď po intranetu či internetu a zajistit ochranu přenášených dat před vnějším zásahem.

Seznam použitých zkratek:

API – Application Programming

BD – Block Diagram

COM – Component Object Model

DA – Data Access

DCOM – Distributed COM

DDE – Dynamic Data Exchange

DLL – Dynamic Link Library

FP – Front Panel

HMI – Human Machine Interface

LV – LabVIEW

NI – National Instruments

OLE – Object Linking and Embeding

OPC – OLE for Process Control

OS – Operační Systém

VI – Virtual Instrument

PLC – Programmable Logic Controller

TCP/IP – Transmission Control Protocol/Internet Protocol

URL – Uniform Resource Locator

Použitá literatura a informační zdroje:

Haasz, V., Rotočil, J., Novák, J. : Číslicové měřicí systémy
ČVUT, Praha 2000, ISBN 80-01-02219-6

Internet: **[1] www.ni.com**
[2] www.opcfoundation.org
[3] www.matricon.com
[4] www.kontron-czech.com
[5] www.wikipedia.org